Meeting with Ben Dorman
Friday October 4, 2002

Attendees: Sandhia Bansal, Dave Davis, Ben Dorman, Heather Kelly, Bob Schaefer

Ben started by asking for some additional information concerning our programming problem. Specifically, he wanted to know why we were using C++ for the science tools, rather than ANSI C. C may be all that is necessary for a series of small tools. Are these tools meant to run independently? C++ is certainly a better C, allowing operator overrides and better organization of large projects. The general consensus was that C++ was a mandate from the LAT team. However, some of the tools will be working together rather than being purely self contained - it is hoped that in these instances we can make use of C++'s strengths.

Dave then asked Ben to explain why they had chosen Tcl/Tk as the scripting language for Xspec. Ben explained that this occurred in 1997, when Python was not as wide spread. There was a great deal of Tcl/Tk experience within LHEA. AIPS++ uses Tk. Tcl/Tk was the only language under consideration that had its own built in widgets. Not Tcl/Tk has object-oriented widget. However, clearly Tcl is not under active development at this time. Meanwhile, Python does not have its own widget library - however many exist. ~~The only problems Ben reported with Tcl/Tk was due to bad locally grown Tcl/Tk code.~~ Xspec11 ports to 4-5 UNIX platforms, but not to windows - due to Fortran code. There were lots of problems with Fortran compiler. XSpec12 plans to support both PGPlot and POW (a locally grown Tk based plotting package).

Ben then went on to explain how they embed Tcl in XSpec12. XSpec itself makes no system calls to Tcl. The XSpec main program looks something like:

```
int main ( ) {
  // initializations
 Tcl_Main(Interp);  // this could start up a GUI or Tcl command line
 return 0;
}

int Tcl_AppInit(interp) {
 XspecInit(interp);  // adds Xspec commands to the Tcl shell
}
```

A STL std::map object is used to ~~created that~~ link~~s~~ names of commands to function pointers. Could initialize other libraries of commands at the same time. Fortran libraries could be handled in a similar fashion.

Starting up Xspec12 produces a Tcl command line, with the Xspec prompt:

```
Xspec>
```

Tcl provides a mechanism to load shared libraries in an operating system independent fashion, by doing:

Xspec> package require [name] [version]
So a specific version of a library can be selected.
This causes an initialize method to be called:

```
int name_Init(interp) {
  // initialize library that has been loaded
}
```

This mechanism of loading shared libraries is useful for adding new options, such as was done for Integral to handle additional data formats.

Tcl command line also provides access to all system level commands. If a command provide to the Tcl interpreter is unrecognized - it is thrown to the system level to see if it exists there, ex.~~)~~ ls.

Tcl also has an eval command which can execute a set of Tcl commands stored in a flat file,

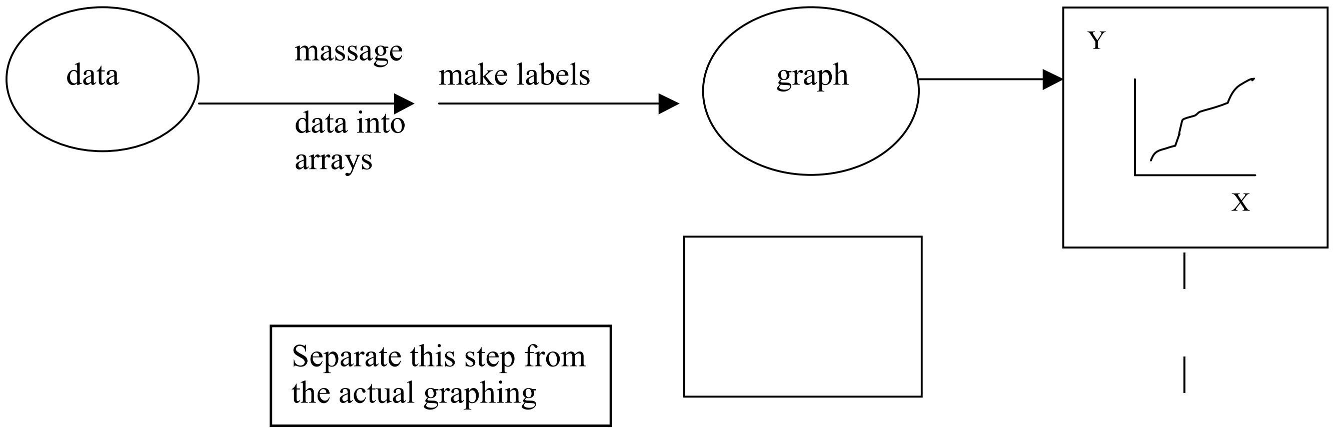
Streams
Ben went on to explain how C++ streams were utilized in Xspec12.
Two classes were created that derive from C++'s standard stream classes:
XSstream derives from iostream and XSstreambuf derives from streambuf.
XSstreambuf contains an XSchannel which reads/writes from/to a stream into a streambuf. The XS~~s~~channel can be an interface to a Tk widget for a GUI or a Tcl channel. In this way, Xspec can handle I/O using the same code for both the GUI and the command line (and the same scheme can clearly be used for any other embedded interpreter: the XSchannel interface is independent of tcl/tk) .

Development and Design
Ben suggested that we have 2 choices:
Xspec12 has identical ~~based its~~ ~~design~~ library dependencies as the ~~on~~ the FITS file viewer, fV - probably the best Ftool available that supports all platforms including Windows. The FITs interface is handled through the fitsTcl library, which adds FITS related commands to the tcl language ~~Fits/Tcl.~~
Or
We can conform to HEADAS and PIL (Parameter interface library, a C language implementation of HEASARC's XPI with enhancements developed by the Integral Science Data Center).

There are no plans to ~~for~~ make Xspec12 conform to the HEADAS interface. ~~to be included in HEADAS~~. HEADAS~~/~~ using PIL does not provide anything that Tcl cannot ~~does~~ not already provide, such as logging (it provides a consistent way of defining default values for parameters which is shared amongst HEADAS tools, but this isn't really relevant to XSPEC. In particular, there is no need to use PIL since Tcl is quitre good at parsing lines, and XSPEC do~~p~~es no~~p~~t use~~d~~ PIL.

Do we want to pipe in/out of the command line science tools? Should this be added to our list of requirements?

Plotting Packages
Ben had some clear suggestions. Specifically, define a clear separation of roles. Xspec11 intertwined its plotting routines with the specifics of its chosen package, pgplot. Xspec12 will support multiple plotting packages and Ben wanted to create a cleaner division of labor.



Ben took a few minutes to explain the advantages of defining shared libraries to add new options. It avoids re-compilation while if-else blocks force recompilation upon addition of a new option. New components are added that derive from a base class and added to a new library. Upon loading the library, the new options are available. This avoids the possibility of breaking existing working code.

We went on to discuss the reasons behind XSpec12 supporting POW. POW was created here and uses the Tk graphics library. Dave pointed out that POW does not produce publication quality plots. Ben asked for any other suggestions for plotting packages. At this point no one is actively supporting POW.

CCFits
CCFits is an interface to cfitsio. Optimized reads, slower writes at this time, will be sped up in future releases. Though the release schedule is unknown at this time. CC~~e~~fits does not yet handle shared memory. At this time, there are problems with the Windows port. Bob offered to install Visual Studio .Net on his machine to allow Ben to opportunity to view the problems and possibly fix them.

Books
Ben recommends the following books:
Herb Sutter's *Exceptional C++*, *More Exceptional C++*

Scott Meyers *Effective C++*, *More Effective C++*
On Ben's computer (veris), browse /data/veris4/dorman/cpp/effective/BOOKINDX/INDEX.HTM
Erich Gamma, et al., *Design Patterns.*
On Ben's computer (veris), browse /data/veris4/dorman/cpp/dpatterns/hires/index.htm